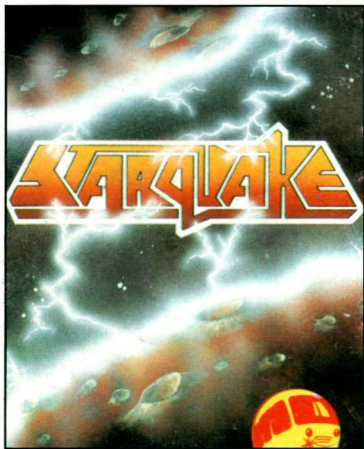
LA SOFTRIVISTA JACKSON PER GLI UTENTI C64&128

JACKSON SOFT compilation

C64&128

NUOVA SERIE ANNO 1 N.1 DICEMBRE 1986 L.8000

SUPERGAME:

STARQUAKE pag. 4

GUIDA ALL'INPUT pag. 7

LISTATO:

DOS-INSPECTOR pag. 8

ARTICOLI:

SALVATAGGIO FRAZIONATO DI PROGRAMMI E I SUOI UTILIZZI pag. 15

LE MANI SUL VIDEO pag. 17

I NUMERI DEI DISPOSITIVI pag. 18

GRUPPO EDITORIALE JACKSON
DIVISIONE PERIODICI

Spedizione in abbonamento postale - Gruppo III - 70

Ogni mese in edicola troverete questa nuova rivista-compilation con cassetta dedicata ai computer C64 e C128.
La cassetta reca un videogame originale, direttamente dall'Inghilterra, scelto tra quelli che si trovano ai vertici delle classifiche. La descrizione del videogame è una recensione accurata e approfondita con i consigli di un superesperto per diventare dei veri campioni.
Oltre al gioco, una serie di listati riguardanti giochi, grafica, utility, da battere direttamente e poi gli articoli più disparati per conoscere a fondo i segreti e i trucchi del vostro computer.
Questo è Jackson Soft Compilation: una pubblicazione unica perché ogni videogame è il meglio che si può trovare sul mercato e le relative recensioni vere e proprie guide al gioco, perché i listati sono autentici, perché gli articoli sono validi.

Il Gruppo Editoriale Jackson, propietario esclusivo dei diritti per l'Italia di questo gioco, invita i lettori che ne fossero a conoscenza, a segnalare l'esistenza di eventuali altre pubblicazioni contenenti questo stesso gioco, alla redazione della nostra rivista. Tali segnalazioni saranno convenientemente compensate.

⭐ SUPERGAME ⭐
STARQUAKE

"Oltre a essere uno dei migliori arcade-adventure da giocare è anche il più carino da vedere con B.L.O.B., il simpatico protagonista dalla spiccata personalità e dallo sguardo tenero". -Zzap 64.
"Quando è stata l'ultima volta che vi siete divertiti con un videogioco? Finalmente ecco la risposta alle preghiere dei giocatori esausti e stufi dei soliti videogames". -Computer & Videogames.
"Lo amiamo. Compratelo e divertitevi!". -Sinclair User.

Queste sono alcune delle positive valutazioni della stampa specializzata inglese su STARQUAKE, il gioco di questo mese della serie Super Soft.
Realizzato inizialmente per lo Spectrum da Stephen Crow, il programmatore di Wizard's Liar, STARQUAKE ha vinto il Golden Joystick Award, premio assegnato annualmente ai videogiochi più meritevoli e, quindi, recentemente è stato tradotto per il Commodore 64 da Nick Strange. Questa nuova versione è più impegnativa, ha un'animazione migliore e il droide protagonista ha un espressione più simpatica.
STARQUAKE è un'arcade-adventure dall'ottima grafica, con un'eccezionale giocabilità e soprattutto un divertente utilizzo di oggetti e personaggi.
Se queste caratteristiche non vi bastano, pensate che il gioco ha anche uno sviluppo di ben 512 schermi e ogni volta è sempre diverso.

La struttura del gioco ricorda i platform della serie Wally Week ormai famosi per i lettori di Oro Soft, oppure i classici realizzati dall'Ultimate. Quest'ultima (scusate il gioco di parole), ha detenuto per molto tempo il monopolio di questo genere di adventure con titoli come Underwurld, Nightshade, Sabrewulf. STARQUAKE è riuscito a interrompere questa egemonia imponendosi come gioco appassionante dal ritmo esasperato, con molti problemi da risolvere, codici da scoprire e soprattutto stupefacente per le numerose sorprese e novità che nasconde.

IL FATTO

Tutto ha inizio quando dallo spazio giunge questo messaggio:
"Un nuovo instabile e non certo dal punto di vista politico, pianeta è stato localizzato nei pressi di un 'Black Hole' (buco nero)".
Questa precarietà è molto pericolosa. Gli studi e le ricerche degli scienziati hanno rilevato che se il centro non verrà ricostruito in fretta garantendo un saldo e definitivo assetto, il pianeta esploderà causando un altro BIG-BANG o terremoto stellare o "starquake", con la conseguente distruzione totale dell'universo.
La missione è suicida e tutti gli eroi conosciuti, sia terrestri che meccanici, si sono rifiutati di andare incontro a una morte sicura. L'unica speranza di salvezza è così garantita dal B.L.O.B. (Bio-Logicaly Operated Being), un particolare droide dalle dimensioni ridotte forse meno abile di altri, ma abbastanza stupido e incosciente per accettare con entusiasmo un simile compito.

L'INIZIO

In compagnia di una fondamentale guida all'universo, il B.L.O.B. si accinge ad atterrare con la sua astronave sul pianeta. Il computer di volo inizia a trasmettere un messaggio:
"Contatto imminente. Preparatevi per la missione Starquake".
CRASH... BANG... SMASH
"Touchdown. Computer malfunction.
Malfunnythinkin.....
Qualcosa non deve essere andato per il verso giusto e infatti il B.L.O.B. si trova sulla superficie del pianeta nei pressi dei rottami dell'astronave e il suo temperamento audace e incosciente inizia a vacillare di fronte a quel tremendo impatto. In ogni caso in questo momento inizia il viaggio all'interno di questa bomba spaziale alla ricerca dei pezzi del centro del pianeta.

LA MISSIONE, IL PIANETA, I NEMICI E... TUTTO IL RESTO

Il pianeta ha una complessa struttura a labirinto costituita da ostacoli naturali come gallerie, caverne, cunicoli e rocce. Come se non fossero sufficienti queste asperità, una fauna molto particolare costituita da alieni dalle forme più strane, giganti pulci, piccoli uccelli appuntiti, girasoli volanti fanno di tutto per 'succhiarvi' energia. Ancora più micidiali sono invece dei manufatti volanti che vi uccidono al minimo contatto.
Quindi la ricerca non è certo agevole e, anche se meno estesa, è più impegnativa e nello stesso tempo divertente di quella di QUO VADIS, il primo e mitico gioco della serie Oro Soft.
Il vostro B.L.O.B. inizia la sua avventura armato di un raggio mortale per le creature aliene. Può solo camminare e generare delle effimere piattaforme che gli permettono con difficoltà di risalire. Questo metodo comunque è valido per delle brevi ascese; per i lunghi spostamenti è consigliabile utilizzare delle speciali piattaforme (SPACE HOPPER) simili a dei tappeti volanti e che trasformano il B.L.O.B. in una specie di cavalletta spaziale e che gli danno anche un'arma più potente.
Se con questo speciale mezzo di trasporto è più facile e sicuro muoversi, è praticamente impossibile raccogliere pezzi del centro del pianeta ed entrare nelle porte oppure nelle piramidi. Una volta raccolta, la speciale piattaforma può essere abbandonata in una delle speciali postazioni sparse in tutto il pianeta.
Un altro mezzo di trasporto è il sistema di teletrasporto o TELEPORT. Ogni cabina è riconoscibile per le speciali antenne sul tetto (la prima la trovate in fondo al primo crepaccio). Entrando in una di queste cabine vi viene comunicato il suo codice e quindi potete spostarvi velocemente in un altra caverna digitando il codice corrispondente.
La maggior parte degli oggetti sono gli equipaggiamenti di riserva che riforniscono di energia il vostro B.L.O.B. e i suoi accessori. Il joystick invece vi fa guadagnare

una nuova vita. Per questi oggetti è sufficiente toccarli per raccoglierli mentre per trasferire nella finestra di trasporto i pezzi del centro del pianeta oppure gli speciali lasciapassare, dovete muovere la leva del joystick in avanti.
Lo speciale lasciapassare o FLEXIBLE THINGYDOOS vi permette di accedere alle piramidi (PYRAMID OF CHEOPS) contenenti 4 possibili pezzi di centro del pianeta. Altre porte speciali (SPACE LOCKS) richiedono chiavi triangolari mentre le porte di sicurezza possono essere superate sempre con il lasciapassare flessibile.
Altre caratteristiche sono i passaggi segreti che vi permettono di attraversare le pareti, gli speciali ascensori anti gravità utilizzabili quando non si hanno le piattaforme fisse e passaggi bloccati da trappole eliminabili con un salto.
In altri punti dovete fare attenzione ai campi di forza intermittenti e mortali generati da speciali elettrodi.

LO SCHERMO

La parte alta del video è occupata dai dati del computer del B.L.O.B.
Da sinistra verso destra avete i seguenti indicatori:
– Il contapunti che assomiglia a un contachilometri.
– Il numero di B.L.O.B. che avete a vostra disposizione. Iniziate ogni avventura con cinque B.L.O.B.
– Tre barre che indicano la quantità di energia del B.L.O.B., delle piattaforme e dell'arma. Quando si esaurisce la prima barra perde te una vita.

– Infine la finestra che indica il pezzo di pianeta raccolto o l'oggetto.

COMANDI

Il B.L.O.B. (Bio-Logically Operated Being) può essere comandato sia con il joystick collegato nella porta 2 che con la tastiera.
Nel primo caso oltre a muovere il B.L.O.B. a destra e a sinistra potete generare le piattaforme muovendo la leva indietro e raccogliere gli oggetti spostandola in avanti. Premendo il pulsante potete sparare per uccidere i vari alieni.
Nel caso utilizziate la tastiera questi sono i comandi:

Z	- SINISTRA
X	- DESTRA
F7	- GIÙ O POSA LA PIATTAFORMA
F5	- ALTO O RACCOGLIE UN OGGETTO
SHIFT	- FUOCO
F1	- PAUSA
RESTORE	- ABBANDONA GIOCO
F3	- ABBANDONA PARTITA
H	- PUNTEGGI RECORD

ALCUNI CONSIGLI

Per completare STARQUAKE e salvare l'Universo potete utilizzare una incredibile varietà di strategie.
Gli ingredienti richiesti per riparare il centro del pianeta variano da gioco a gioco. Fortunatamente, come vi abbiamo già detto, ci sono extra live e supplementi di energia in numero superiore a quelli delle avventure di Wally Week.
Quindi la mappa del pianeta è sempre la stessa e ogni gioco varia per la disposizione degli oggetti bonus e dei vari pezzi da recuperare per ricostruire il centro del pianeta.
Il primo consiglio è quello di disegnare la mappa annotando la locazione delle varie teleporte con i relativi codici, delle serrature, delle piattaforme e dei passaggi segreti.
Il centro del pianeta è formato da 9 pezzi anche se quelli sparsi sul pianeta sono più numerosi. Recatevi il più presto possibile al centro per verificare quali sono gli oggetti da raccogliere ed evitare così un lavoro inutile.
Fondamentali sono i lasciapassare che vi permettono di entrare nelle piramidi e avere l'accesso ad altre porte di sicurezza. Posizionate il B.L.O.B. sopra la piramide e spostate la leva in avanti. Una volta all'interno scambiate gli

oggetti inutili con eventuali pezzi del pianeta. Per depositarli al centro è sufficiente raggiungere lo schermo che si trova sulla destra di quello con le tre torri. Se il pezzo o i pezzi sono quelli giusti si sistemeranno automaticamente. Fate attenzione a quando vi lasciate cadere a peso morto nei cunicoli perché potreste infilzarvi in qualche ostacolo mortale e soprattutto non rimanete senza energia per le piattaforme in qualche buca o in fondo a qualche cunicolo. Non avreste scampo.
Molto difficile è difendersi quando generate le piattaforme, quindi utilizzate il più possibile quelle fisse.
Evitate di utilizzare gli oggetti-energia quando siete al massimo con i livelli. Sprechereste vitale energia che potrebbe esservi utile più avanti.
Per superare le Smash Trap è sufficiente salire di poco con la piattaforma e quindi lasciarsi cadere.
Per ora è tutto, buona fortuna e.... che il joystick sia con voi!

SUPERGAME
STARQUAKE

ISTRUZIONI PER IL CARICAMENTO

Introdotta la cassetta nel registratore e con il nastro riavvolto all'inizio premete contemporaneamente i tasti SHIFT e RUN/STOP sul vostro computer.
Quando appare la scritta sullo schermo: "Press play on tape" premete il tasto play sul registratore. Il programma verrà caricato automaticamente e in modo veloce.

GALAXY A-Z

B.L.O.B.	*TELETRASPORTATORI*
CAMMINARE	*PIANTE*
VOLARE	*CHIODI*
LASCIARE	*VEGETAZIONE LUNARE*
TASTO PAUSA	*ENERGIA PER PIATTAFORME*
ASCENSORI ANTI-GRAVITÀ	*CARTA CON CODICE D'ACCESSO*
STELLE	*PEZZI DEL CENTRO*
ARMA LASER	*NUVOLE AD ALTA DENSITÀ*
CENTRO PIANETA	*PAESAGGIO LUNARE ROCCIOSO*
ELETTRONI	*STRUTTURA MOLECOLARE*
SEGNALI DI DIREZIONE	*GRUPPO ENERGIA*
TASTO D'ABBANDONO	*POSTAZIONE PER PIATTAFORMA*
LASCIAPASSARE FLEXI	*OGGETTI ANTIMATERIA*
PIRAMIDI DI CHEOPE	*SERRATURE SPAZIALI*
JOYSTICK	*PIATTAFORMA PERMANENTE*
SUONO SPAZIALE	*PIATTAFORMA PROVVISORIA*
TASTIERA	*WEAPON PACKS*
EFFETTI SPECIALI	*BONUS LIVES*
ASTRONAVE	*SCHELETRASTRONAUTI*
PASSAGGI SEGRETI	*TRAPPOLE A SPINTA*
FIORI	*FUNGHI*

GUIDA ALL'INPUT C64-C128

TABELLA DI CONVERSIONE

{HOME} .... HOME
{CLR} .... PULIZIA SCHERMO
{CUR.SU} ...... CURSORE IN ALTO
{CUR.GIU} .... CURSORE IN BASSO
{CUR.DES} .... CURSORE A DESTRA
{CUR.SIN} ..... CURSORE A SINISTRA
{SPC} .... SPAZIO
{RVS ON} ...... REVERSE ON
{RVS OFF} ..... REVERSE OFF
{INST} .... INSERT
{F1} .... TASTO F1
{F2} .... TASTO F2
{F3} .... TASTO F3
{F4} .... TASTO F4
{F5} .... TASTO F5
{F6} .... TASTO F6
{F7} .... TASTO F7
{F8} .... TASTO F8
{BLACK} .... COL. NERO (CTRL+1)
{WHITE} .... COL. BIANCO (CTRL+2)
{RED} .... COL. ROSSO (CTRL+3)
{CYAN} .... COL. CIANO (CTRL+4)
{PURPLE} .... COL. PORPORA (CTRL+5)
{GREEN} .... COL. VERDE (CTRL+6)
{BLUE} .... COL. BLU (CTRL+7)
{YELLOW} ..... COL. GIALLO (CTRL+8)
{ORANGE} ..... COL. ARANCIO (CBM+1)
{BROWN} ..... COL. MARRONE (CBM+2)
{LT. RED} .... COL. ROSSO CHIARO (CBM+3)
{GRAY 1} .... COL. GRIGIO 1 (CBM+4)
{GRAY 2} .... COL. GRIGIO 2 (CBM+5)
{LT. GREEN} . COL. VERDE CHIARO (CBM+6)
{LT. BLUE} ..... COL. BLU CHIARO (CBM+7)
{GRAY 3} .... COL. GRIGIO 3 (CBM+8)

NORME PER LA BATTITURA

I caratteri grafici, ottenuti con la pressione dei tasti "Shift" e "CBM", sono codificati in modo da indicare il tasto da premere assieme a "Shift" o "CBM". Es. il cuoricino è codificato con >SHS<. Il numero dentro le parentesi indica le volte che il tasto va premuto.

DOS-INSPECTOR

di L. Pampana-Biancheri

Approfondimenti, trucchi e piccoli segreti dei drive Commodore (1541 e 2031LP in particolare), il tutto corredato da un programma per curiosare nella loro memoria.

I drive Commodore sono periferiche "intelligenti". Questo significa che dispongono internamente di un vero e proprio micro-computer nella cui ROM è contenuto il DOS (Disk Operating System), cioè l'insieme di routine che gestiscono il drive stesso. Il computer ha il solo compito di inviare al DOS i comandi appropriati (e naturalmente i dati!), senza doversi occupare delle operazioni di controllo del disco. Lo scambio di informazioni con il drive avviene tramite un'opportuna interfaccia, la IEEE-488 (o la sua versione seriale nel caso del C64).

Con il BASIC 4.0 sono disponibili alcune istruzioni che automaticamente trasmettono ordini al DOS, mentre con il BASIC 3.0 (quello del PET 3032 e del C64) i comandi stessi devono essere inviati a cura dell'utente, tramite – come è noto – un file aperto sull'indirizzo secondario 15. Tuttavia è bene notare che tutto quello che si riesce a far fare al drive con il BASIC 4.0 può essere ottenuto anche con il BASIC 3.0: infatti, come si è detto, il DOS risiede nel drive non nel computer!

Questo può sembrare scontato, ma ci sono molte pubblicazioni in cui si afferma il contrario, precisamente quanto erroneamente. per esempio sul manuale del 2031LP si dice che i file relative possono essere gestiti solo con DOS 2 o successivi (e fin qui tutto bene) e BASIC 4.0. Per fortuna l'errore non è stato ripetuto sul manuale nel 1541, ma anche in questo non si fa cenno ad alcune possibilità presenti operando con il BASIC 4.0, che quindi debbono essere accessibili anche con il BASIC del C64 (e del PET 3032).

Di seguito verranno esposte queste possibilità "nascoste", dopo una breve revisione della gestione dei file relative con BASIC 3.0 in cui si mettono in luce le equivalenze con le istruzioni del BASIC 4.0. Poi si parlerà di una possibilità del DOS di cui non c'è traccia in alcun manuale. Quindi si tratterà della mappa di memoria del drive e del modo di accedere a tale memoria, presentando uno strumento software che permetterà di curiosare nella memoria di ogni modello Commodore. Infine si farà cenno alla possibilità di far eseguire dal microprocessore del drive delle routine scritte dall'utente.

FILE RELATIVE, APPEND E CONCAT CON BASIC 3.0

Un file relative è costituito da un insieme di record di lunghezza prefissata, a differenza di quanto accade per i sequenziali in cui il record ha la lunghezza dell'informazione che deve effettivamente contenere. Questo svantaggio in termini di spazio occupato sul dischetto è compensato dalla possibilità di avere un accesso casuale al record voluto (o addirittura, all'interno del record, al singolo campo) per poter indifferentemente leggere o scrivere nel file.

Vediamo in pratica come si crea un file relative con le due versioni del BASIC. Con il BASIC. 4.0 si ha
DOPEN# nf, "nome",D (dr),L(r len),U(dn)
mentre con il BASIC 3.0 OPEN15,dn,15:OPEN nf,dn,ch,"dr:nome,L,"+CHR$(r len)
essendo
nf= numero del file logico
dn= numero di device
ch= indirizzo secondario (fra 2 e 14)
dr= numero drive
r len= lunghezza record (fra 1 e 254)
Si noti che r len deve includere anche i terminatori di ogni campo. Se il file esiste già questa indicazione si può omettere.

Per accedere al file occorre posizionare il puntatore al record voluto e, all'interno del record stesso, all'inizio del campo che si vuole leggere o in cui si vuole scrivere.

In BASIC 4.0 la sintassi è:
RECORD# nf, (nr),(b)
mentre in BASIC 3.0
PRINT# 15,"P" CHR$ (96+ch) CHR$ (nrhi) CHR$ (nr lo) CHR$ (b)
essendo nr=
numero record voluto (fra 1 e 720 per il DOS 2.6)
nrhi= INT (nr/256)
nrlo= nr-nrhi*256
b= byte del record a cui inizia il campo voluto (se non indicato si assume 1, primo byte del record).
A questo punto se il record indicato esiste già si potrà leggerlo o modificarlo (con le solite istruzioni PRINT#nf, dati INPUT#nf, variabili e GET#nf, va-

riabile) mentre in caso che il record non esista ancora si avrà un errore di tipo 50 (record not present) che ci avverte che nel record si può solo scrivere e non leggere.
Alla fine del lavoro il file relative viene chiuso nel modo solito: in BASIC 4.0 con
DCLOSE#nf
in BASIC 3.0 con
CLOSE nf: CLOSE 15
Come si può vedere le differenze fra le due versioni di BASIC non hanno nulla di concettuale: si tratta solo di una diversa forma delle istruzioni da usare.
Per quanto riguarda i file sequenziali (siano essi di tipo SEQ, PRG o USR) il BASIC 4.0 (con DOS 2) dispone di due interessanti opportunità: l'APPEND, cioè la possibilità di aprire un file per aggiungervi dati, senza doverlo riscrivere da capo, e il CONCAT, cioè la possibilità di copiare un file in coda a un altro.
Dai manuali (neppure da quello del 1541, che invece espone l'uso dei file relative con il BASIC 3.0) non risulta affatto che queste possibilità siano, come invece si verifica, accessibili anche con il BASIC 3.0.
La sintassi BASIC 4.0 è, nel primo caso
APPEND#nf,"nome",D(dr),U(dn)
In BASIC 3.0 è sufficiente che, aprendo normalmente il file, si indichi A invece dei soliti R o W:
OPEN nf, dn,sa, "dr:nome, tipo, A"
Per il concatenamento abbiano in BASIC 4.0
CONCAT D(dr 2), "secondo" TO D(dr1), "primo" ON U(dn)
mentre in BASIC 3.0:
OPEN 15,dn,15
PRINT#15,"C@ dr1:primo=dr1:primo,dr2:secondo"

UN SEGRETO DEL DOS 2.6

Il DOS 2.6 ha una possibilità che non viene assolutamente menzionata nei manuali; è stata riscontrata su 1541 e 2031LP ma forse vale anche per altri modelli o versioni del DOS: per scoprirlo basta provare.
Un file sequenziale (di tipo SEφ, PRG o USR) può essere aperto in lettura (L) o in scrittura (W). Si è visto sopra che può anche essere aperto per aggiungere dati (A), ma esiste una quarta possibilità.
È noto che dopo aver aperto un file in scrittura occorre chiuderlo prima di rimuovere il dischetto o spegnere il drive o fare operazioni quali il caricamento di un programma, l'initialize, il validate, eccetera. Altrimenti il contenuto del buffer di dati non viene salvato sull'ultimo blocco del file, che quindi conterrà tutt'altro. In tale funesto caso il file nella directory appare con un asterisco a fianco del tipo. Se si tentasse di accedervi in lettura o in append si otterrebbe l'errore 60 (write file open), in scrittura l'errore 63 (file exist). Pare dunque che il file sia definitivamente perduto: non rimane che cancellarlo (e fare il validate perché in questi casi di solito qualche blocco resta allocato).
Si pensi al caso in cui, avendo a che fare con un file importante, fosse mancata la corrente! Una bella sfortuna: tutti i dati sarebbero persi. Invece c'è il modo di recuperarli, almeno fino all'ultimo blocco, quello famigerato che era ancora nel buffer del drive quando è mancata la corrente.
Basta aprire il file normalmente ma usando M come modo (anziché R, W o A): in tal caso il DOS non si oppone all'apertura (che considera come se fosse in lettura) anche se il file non è chiuso correttamente:
OPEN nf, dn, ch, "dr:nome, tipo, M"
Si noti però che dobbiamo trovare noi il modo di accorgerci di quando siamo arrivati alla fine dei dati validi: l'ultimo blocco del file, infatti, non è stato scritto sul dischetto e quindi conterrà dati estranei. Inoltre il puntatore di questo blocco punterà chissà dove: se abbiamo fortuna a un numero di traccia o settore illegali (e avremo l'errore 66, illegal track and sector), ma forse al blocco di un vecchio file cancellato.
Quindi saremo noi a dover decidere dove arrestarci nella lettura, discriminando fra dati validi e no. Naturalmente ci sono casi in cui questo è più facile, a seconda del contenuto del file. Di solito comunque è meglio che cominciare da capo!
La cosa più semplice è scrivere un programma che mostri ogni byte assieme al suo numero d'ordine, in modo da poter individuare fino a che punto i dati sono validi, e un altro che recuperi il numero di byte da noi indicati. Si vedano a tale scopo le figure 1 e 2.
Si potrebbe addirittura, in fase di scrittura, prevedere il salvataggio premettendo a ogni record un byte che contenga il checksum del record stesso. In tal caso il recupero dopo un eventuale incidente diventerebbe automatico: basterebbe fermarsi quando il record letto non ha il checksum corrispondente al byte iniziale (ricordarsi di rileggere i record con dei GET# per non avere l'errore "string too long" alla fine dei dati validi!).

LA MEMORIA DEI DRIVE

Come si è detto sopra, i drive Commodore sono costituiti da un vero e proprio micro-computer. Quindi hanno una loro RAM in cui conservano i dati che servono al micro-processore per lavorare, e in cui risiedono i bufferro di 256 byte dai quali passano i dati che verranno letti o scritti sul dischetto un blocco alla volta.
Per la verità i modelli "professionali" dispongono addirittura di due processori che hanno in comune una parte di RAM: uno di essi si occupa del colloquio con l'interfaccia (IP, Interface Processor) e uno del trasferimento dei dati da e per la superficie del disco (FDC, Floppy Disk Controller). In ogni caso si tratta di micro-processori della famiglia 6500, dotati dello stesso set di istruzioni del 6502. Nella figura 3 è riportata la mappa di memoria del drive 1541, uguale a quella del 2031LP (cambia solo – e in parte – il contenuto della ROM): entrambi questi modelli, oltre ad avere meno RAM

(quindi meno buffer) dei modelli professionali, dispongono di un solo processore (un 6502) che, con tecniche di interrupt, svolge entrambi i compiti. Questo spiega perché sono più lenti degli altri modelli (non spiega però perché il 1541 è molto più lento del 2031LP: il fatto che l'interfaccia sia seriale non pare un motivo sufficiente data la notevole differenza).

Ma vediamo come si accede alla memoria del drive (quanto detto di seguito vale per tutti i modelli; nel caso di quelli con doppio processore i comandi si riferiscono alla memoria vista dall'IP).

Per leggere un byte invieremo sul canale di controllo (che supporremo aperto con OPEN 15,dn,15) la seguente istruzione:

PRINT# 15, "M-R"CHR$(addr lo)CHR$(addrhi)

essendo

addr lo= parte bassa di addr (indirizzo da leggere) = INT(addr/256)

addrhi=addr-256*Baddrhi

Quindi il byte verrà letto sul canale di errore (prima di effettuare ogni altra operazione):

GET#15,A$:A=ASC(A$+CHR$(0))

Per inviare una sequenza di byte alla memoria (fino a 34 per volta) invece si invierà l'istruzione

PRINT#15, "M-W"CHR$ (addr lo)CHR$ (addrhi)CHR$(n)CHR$(data1)... ...CHR$(datan)

essendo

n= numero byte da inviare

data1...datan= byte da inviare

Queste istruzioni sono usate nel programma da cui questo articolo prende nome, che permette di indagare nella memoria di qualunque modello di drive con qualunque modello di computer Commodore (con il VIC 20 ci saranno problemi sia di memoria disponibile che di output sullo schermo).

Con DOS-INSPECTOR è possibile fare il dump, cioè visualizzare il contenuto della memoria (codici esadecimali e rispettivo carattere ASCII), nonché alterarlo (attenti però...) nonché disassemblare il DOS. Il dump e il disassemblaggio possono anche essere diretti su stampante.

Il programma, scritto in BASIC, non è particolarmente veloce, ma in effetti se l'out è diretto su stampante è la sua velocità di stampa che limita in pratica quella del programma stesso, mentre se è diretto su schermo sarebbe inutile "sparare" dati a una velocità di gran lunga superiore alle possibilità di lettura umane! Comunque chi volesse un programma più veloce può sempre servirsi di un compilatore BASIC (già con l'Austro Compiler si guadagna un 50%).

ESECUZIONE DI ROUTINE NELLA MEMORIA DEL DRIVE

Fino a ora abbiamo visto come introdurre dati nella RAM del drive e come esaminare il DOS disassemblandolo.

A questo punto con un po' di buona volontà è possibile comprendere a fondo il funzionamento del drive stesso, e quindi al limite scrivere delle routine in l.m. che gli permettano di eseguire compiti non previsti dal DOS (o di eseguirli in modo diverso, come fanno per esempio tutti i copiatori veloci scritti per il 1541).

I comandi che ordinano al processore del drive di eseguire le routine scritte dall'utente sono diversi.

Il più semplice è il seguente:

PRINT#15,"M-E"CHR$(addr lo)CHR$(addrhi)

che fa partire l'esecuzione dalla locazione addr.

Con le istruzioni

PRINT#15,"U3"

....

PRINT#15,"U8"

si eseguono le routine che iniziano (per il 1541 e 2031LP) alle locazioni rispettivamente $0500, $0503, $0506, $0509, $050C, $050F.

In ogni caso, a meno che non si vogliano eseguire routine già presenti sulla ROM, occorrerà aprire un file random riservando a esso il buffer in cui immetteremo le nostre routine, in modo da evitare che il DOS vada a metterci altri dati. Questo si ottiene, come è noto, con

OPEN nf, dn.ch, "#nbuffer"

rilevando dal canale di errore se il buffer richiesto (nbuffer) è disponibile (in caso contrario si ottiene un errore 70, no channel).

Avendo aperto il file random è possibile servirsi di un altro mezzo di esecuzione di routine: con

PRINT#15,"B-E"ch;dr;track;sector

il drive legge dal disco il blocco indicato da track e sector e lo mette nel buffer riservato al file random aperto con indirizzo secondario ch; poi lo esegue a partire dall'inizio.

Ma c'è un'ultima possibilità tenuta ben nascosta dai manuali (è stata riscontrata sia sulla 2031LP che sul 1541, ma può darsi che valga anche per altri modelli).

Si tratta dell'istruzione

PRINT#15,"&nome"

Essa provoca il caricamento nella RAM del drive e l'esecuzione (nel modo che vedremo) del file (di tipo SEQ, PRG o USR) che ha nome appunto "&nome".

La struttura di "&nome" è piuttosto complessa: anzitutto è possibile che questo file contenga uno di seguito all'altro più blocchi (di lunghezza variabile) di istruzioni da caricare in punti diversi della RAM del drive. In ogni caso l'esecuzione partirà dall'indirizzo di caricamento del primo di questi blocchi.

Ogni blocco avrà la struttura seguente:

1 - parte bassa dell'indirizzo di caricamento
2 - parte alta dell'indirizzo di caricamento
3 - numero n di byte da caricare
4 - primo byte da caricare
....
n+3 - ultimo byte da caricare
n+4 - checksum del blocco

Il checksum del blocco viene calcolato sommando assieme, uno dopo l'altro, tutti i byte del blocco (inclusi l'indirizzo di caricamento e il n. di byte ed escluso naturalmente il checksum stesso) e sottraendo 255 ogni volta che il risultato di una somma supera 255.

Come si vede questa possibilità del DOS è piuttosto potente in quanto l'utente, una volta scritto il programma &nome, per farlo eseguire non dovrà far altro che dare il comando corrispondente, senza neppure dover scrivere un programma BASIC che carichi i dati né doverli mettere per forza in un determinato blocco del dischetto!

REMark programma DOS-INSPECTOR

1070 - viene aperto il canale di controllo.
1080 - vengono dimensionati i vettori che conterranno rispettivamente i simboli esadecimali, gli mnemonici, gli indirizzamenti e i loro indicatori mnemonici.
1090-1140 - vengono caricati i vettori suddetti.
1170-1220 - menu principale.
1230 - salto all'esecuzione del dump o del disassemblaggio.
1240-1410 - modifica della memoria del drive.
1420-1500 - poke nella memoria del drive.
1520-1560 - conversione decimale/esadecimale di un numero fra 0 e 255.
1570-1630 - conversione decimale/esadecimale di un numero fra 0 e 65535.
1640-1820 - conversione esadecimale/decimale di un numero fra $0 e $FFFF.
1830-1970 - parte comune al dump e al disassemblaggio: gestisce la scelta fra stampante e schermo e l'input dell'indirizzo di partenza.
1980-2090 - dump della memoria del drive.
2100-2170 - peek nella memoria del drive.
2180-2680 - disassemblaggio.
2220 - inizializza il contatore delle linee visualizzate, per potersi fermare quando completa la pagina su video.
2310 - byte sconosciuto: non è un'istruzione valida dell'assembly 6502.
2320 - sceglie gli mnemonici dell'indirizzamento.
2330-2340 - sceglie la routine appropriata per gestire la visualizzazione dell'operando.
2690-3170 - elenco degli mnemonici del 6502 (incluso per gli indirizzamenti accumulatore e immediato la A o il #) seguiti ognuno dal rispettivo indirizzamento codificato come segue:
implicito e accumulatore: 0
assoluto: 30
pag. zero: 20
immediato: 10
assoluto, x: 31
assoluto, y: 32
(indiretto, x): 23
(indiretto), y: 24
pag. zero, x: 21
relativo: 40
indiretto: 35
pag. zero, y: 22
Si noti che la prima cifra si riferisce al modo di visualizzazione dell'indirizzo, mentre la seconda indica lo menemonico dell'indirizzo stesso.

Si noti infine che per codici illegali è immessa una coppia di stringhe nulle: il read di una stringa nulla in una variabile numerica (intera o reale che sia) dà come risultato 0.

NOTA: in tutti i listati sono presenti, alla fine di ogni linea, le REM con i checksum calcolati con il programma "Checksum 64" di Ercole Colonnese, apparso a pag. 183 di Bit n. 59 (marzo 1985). Si faccia riferimento a tale articolo per l'utilizzazione dei checksum in fase di inserimento del listato. Chi non disponesse del programma menzionato può semplicemente ignorare le REM in fine di linea, senza inserirle.

④ *Listato del programma DOS-INSPECTOR.*

Il programma è destinato a indagare nella memoria del drive, ma è possibile modificarlo (ved. fig. 7) per curiosare nella memoria del computer stesso. Volendo, DOS-INSPECTOR può essere compilato, per esempio con AUSTRO-COMPILER o PET-SPEED. Al termine di ogni linea è presente una REM seguita dal checksum della linea stessa: tali REM non devono comparire nel listato definitivo in quanto servono solo a controllare che la linea inserita sia corretta. Per il loro utilizzo si veda la nota in fondo alle REMark.

```
1000 REM ******************************
1010 REM *                            *
1020 REM * D O S - I N S P E C T O R *
1030 REM *                            *
1040 REM ******************************
1050 REM
1060 PRINT"{CLR}{6 CUR.DES}{CUR.GIU}{RVS
      ON} D O S - I N S P E C T O R {2 C
     UR.GIU}"
1070 OPEN15,8,15
1080 DIMEX$(15),C$(255),C%(255),NA$(5),N
     B$(5)
1090 FORI=0TO9:EX$(I)=RIGHT$(STR$(I),1):
     NEXT
1100 EX$(10)="A":EX$(11)="B"
1110 EX$(12)="C":EX$(13)="D":EX$(14)="E"
     :EX$(15)="F"
1120 FORI=0TO255:READC$(I),C%(I):NEXT
1130 NA$(0)="":NA$(1)="":NA$(2)="":NA$(3
     )="(":NA$(4)="(":NA$(5)="("
1140 NB$(0)="":NB$(1)=",X":NB$(2)=",Y":N
     B$(3)=",X)":NB$(4)="),Y":NB$(5)=")"
1150 PRINT"{CLR}{6 CUR.DES}{CUR.GIU}{RVS
      ON} D O S   I N S P E C T O R {2 C
     UR.GIU}"
1160 CLOSE1:OPEN1,3:R$="V"
1170 PRINT"{RVS ON}L{RVS OFF}ETTURA, {RV
     S ON}M{RVS OFF}ODIFICA,":PRINT"{RVS
      ON}D{RVS OFF}ISASSEMBLAGGIO O {RVS
      ON}F{RVS OFF}INE ? ";
1180 GETJ$:IFJ$<>"L"ANDJ$<>"M"ANDJ$<>"D"
     ANDJ$<>"F"THEN1180
1190 PRINTJ$
1200 IFJ$="F"THENCLOSE1:END
1210 PRINT"{CUR.GIU}N.B.: ACCETTA ANCHE
     INDIRIZZI DI"
1220 PRINT"PARTENZA ESADECIMALI SE PRECE
```

```
     DUTI DA $."
1230 IFJ$="L"ORJ$="D"THENPRINT"{CUR.GIU}
     S=STAMPANTE; V=VIDEO; M=MENU":GOTO1
     860
1240 REM
1250 REM *** MODIFICA ***
1260 REM
1270 INPUT"{CUR.GIU}IND. DI PARTENZA (OP
     PURE {RVS ON}M{RVS OFF}ENU)=";A$
1280 IFA$="M"THEN1150
1290 IFLEFT$(A$,1)="$"THENGOSUB1670:GOTO
     1310
1300 PC=VAL(A$)
1310 MC=PC:IFMC>65535THEN1270
1320 PRINT"METTI M NELLA PRIMA CASELLA P
     ER TORNARE"
1330 PRINT"AL MENU.":PRINT
1340 NN=MC:GOSUB1570:PRINT:PRINTNN$"{3 S
     PC}";:S$=""
1350 FORJ=1TO8:PC=MC:GOSUB2130:GOSUB1550
     :S$=S$+N$+" ":MC=MC+1:NEXT
1360 PRINTS$:INPUT"{CUR.SU}{6 CUR.DES}";
     RR$
1370 IFLEFT$(S$,23)=RR$THENPRINT"{CUR.SU
     }";:GOTO1340
1380 IFLEFT$(RR$,1)="M"THEN1270
1390 MC=MC-8
1400 PRINT"{CUR.SU}{10 CUR.DES}";:FORJ=1
     TO8:A$="$"+MID$(RR$,3*J-2,2)
1410 GOSUB1670
1420 REM
1430 REM *** POKE  NELLA  MEMORIA ***
1440 REM *** DEL DRIVE E VERIFICA ***
1450 REM
1460 MH=INT(MC/256):ML=MC-MH*256
1470 PRINT#15,"M-W"CHR$(ML)CHR$(MH)CHR$(
     1)CHR$(PC)
1480 PRINT#15,"M-R"CHR$(ML)CHR$(MH):GET#
     15,X$:IFX$=""THENX$=CHR$(0)
1490 X=ASC(X$):IFX<>PCTHENPRINT"?{CUR.SI
     N}";
1500 MC=MC+1:PRINT"{3 CUR.DES}";:NEXT
1510 GOTO1340
1520 REM
1530 REM *** CONVERSIONE DEC/HEX ***
1540 REM
1550 N$=EX$(INT(ND/16))+EX$(ND-INT(ND/16
     )*16)
1560 RETURN
1570 ND=INT(NN/256)
1580 GOSUB1550
1590 NN$=N$
1600 ND=NN-INT(NN/256)*256
1610 GOSUB1550
1620 NN$="$"+NN$+N$
1630 RETURN
1640 REM
1650 REM *** CONVERSIONE HEX/DEC ***
1660 REM
1670 IFLEN(A$)>5THENPC=0:RETURN
1680 FORI=1TO4
1690 A$(I)="0":NEXT
1700 FORI=2TOLEN(A$)
1710 A$(LEN(A$)-I+1)=MID$(A$,I,1)
1720 NEXT
1730 FORI=1TO4
1740 IFA$(I)="A"THENA$(I)="10"
1750 IFA$(I)="B"THENA$(I)="11"
1760 IFA$(I)="C"THENA$(I)="12"
1770 IFA$(I)="D"THENA$(I)="13"
1780 IFA$(I)="E"THENA$(I)="14"
1790 IFA$(I)="F"THENA$(I)="15"
1800 NEXT
1810 PC=INT(4096*VAL(A$(4))+256*VAL(A$(3
     ))+16*VAL(A$(2))+VAL(A$(1)))
1820 RETURN
1830 REM
1840 REM *** COMUNE A  LETTURA ***
1850 REM ***  DISASSEMBLAGGIO  ***
1860 REM
1870 PRINT"{CUR.GIU}OUTPUT SU ";:IFR$="V
     "THENPRINT"VIDEO;":GOTO1890
1880 PRINT"STAMPANTE;"
1890 INPUT"IND. DI PARTENZA (OPPURE S/V/
     M)=";A$
1900 IFA$="V"THENR$=A$:CLOSE1:OPEN1,3:GO
     TO1860
1910 IFA$="S"THENR$=A$:CLOSE1:OPEN1,4:GO
     TO1860
1920 IFA$="M"THEN1150
1930 IFLEFT$(A$,1)="$"THENGOSUB1670:GOTO
     1950
1940 PC=VAL(A$)
1950 IFPC>65535THEN1860
1960 PRINT"PREMI UN TASTO QUALUNQUE PER
     FERMARE.{CUR.GIU}"
1970 IFJ$="D"THEN2220
1980 REM
1990 REM *** LETTURA ***
2000 REM
2010 NN=PC:GOSUB1570:PRINT#1:PRINT#1,RIG
     HT$(NN$,4)"{2 SPC}";
2020 FORI=1TO8
2030 GOSUB2130:AS(I)=ND:GOSUB1550:PRINT#
     1,N$" ";:PC=PC+1:NEXT
2040 PRINT#1," ";:FORI=1TO8
2050 IF(AS(I)>31ANDAS(I)<128)OR(AS(I)>15
     9)THENPRINT#1,CHR$(AS(I));:GOTO2070
2060 PRINT#1," ";
2070 NEXT
2080 GETAA$:IFAA$<>""THENPRINT#1:GOTO186
     0
2090 GOTO2010
2100 REM
2110 REM *** PEEK NELLA MEMORIA ***
2120 REM ***      DEL DRIVE     ***
2130 REM
2140 PH=INT(PC/256):PL=PC-PH*256
2150 PRINT#15,"M-R"CHR$(PL)CHR$(PH)
2160 GET#15,MC$
2170 ND=ASC(MC$+CHR$(0)):RETURN
2180 REM
2190 REM *** DISASSEMBLAGGIO ***
2200 REM
2210 IFPC<0ORPC>65529THEN2440
2220 RI=1
2230 GETAA$:IFAA$<>""THEN1870
2240 PRINT#1,RIGHT$("{6 SPC}"+STR$(PC),5
     );SPC(1);
2250 NN=PC:GOSUB1570
2260 PRINT#1,NN$;
2270 PRINT#1,SPC(2);
2280 GOSUB2130
2290 GOSUB1550:PRINT#1,N$;
2300 C$=C$(ND):C=C%(ND)
2310 IFC$=""THENPRINT#1,SPC(8);"???":GOT
     O2350
2320 CC=C-INT(C/10)*10
2330 ONC/10GOSUB2440,2490,2540,2610
2340 IFC=0THENGOSUB2430
2350 PC=PC+1:RI=RI+1:IFRI>24THEN2380
2360 IFPC>65529THEN1870
2370 GOTO2230
2380 IFR$="S"THENRI=1:GOTO2230
2390 PRINT"PER FERMARE 'S', PER CONT. 'S
     PACE'.";
```

```
2400 GETA$:IFA$<>"S"ANDA$<>" "THEN2400
2410 IFA$="S"THENPRINT"":GOTO1870
2420 PRINT"":RI=1:GOTO2240
2430 PRINT#1,SPC(8);C$:RETURN
2440 PRINT#1,SPC(1);:PC=PC+1
2450 GOSUB2130
2460 GOSUB1550
2470 PRINT#1,N$;SPC(5);C$;" ";"$";N$
2480 RETURN
2490 PRINT#1,SPC(1);:PC=PC+1
2500 GOSUB2130
2510 GOSUB1550
2520 PRINT#1,N$;SPC(5);C$;" "NA$(CC)"$"N
     $;NB$(CC)
2530 RETURN
2540 PRINT#1,SPC(1);:PC=PC+1
2550 GOSUB2130
2560 GOSUB1550:N2$=N$
2570 PRINT#1,N$;SPC(1);:PC=PC+1
2580 GOSUB2130:GOSUB1550
2590 PRINT#1,N$;SPC(2);C$;" "NA$(CC)"$";
     N$;N2$;NB$(CC)
2600 RETURN
2610 PRINT#1,SPC(1);:PC=PC+1
2620 GOSUB2130:GOSUB1550
2630 PRINT#1,N$;SPC(5);C$;" ";
2640 IFND>127THENNN=PC+ND-255
2650 IFND<127THENNN=PC+ND+1
2660 GOSUB1570
2670 PRINT#1,NN$
2680 RETURN
2690 REM
2700 REM *** ISTRUZIONI MNEMONICHE ***
2710 REM *** ASSEMBLY 6502  E LORO ***
2720 REM ***      INDIRIZZAMENTO    ***
2730 REM
2740 DATABRK,0,ORA,23,,,,,,,ORA,20
2750 DATAASL,20,,,PHP,0,ORA #,10,ASL A,0
     ,,,,
2760 DATA ORA,30
2770 DATAASL,30,,,BPL,40,ORA,24,,,,,,
2780 DATAORA,21,ASL,21,,,CLC,0,ORA,32,,,
     ,
2790 DATA,,ORA,31,ASL,31,,,JSR,30
2800 DATAAND,23,,,,,BIT,20,AND,20,ROL,20
     ,,
2810 DATAPLP,0,AND #,10,ROL A,0,,,BIT,30
2820 DATAAND,30,ROL,30,,,BMI,40,AND,24,,
     ,,,,
2830 DATAAND,21,ROL,21,,,SEC,0
2840 DATAAND,32,,,,,,,AND,31,ROL,31,,,RT
     I,0
2850 DATAEOR,23,,,,,,,EOR,20
2860 DATALSR,20,,,PHA,0,EOR #,10,LSR A,0
     ,,,JMP,30
2870 DATAEOR,30,LSR,30,,,BVC,40,EOR,24
2880 DATA,,,,,,EOR,21,LSR,21,,,CLI,0
2890 DATAEOR,32,,,,,,,EOR,31,LSR,31
2900 DATA,,RTS,0,ADC,23,,,,,,,ADC,20
2910 DATAROR,20,,,PLA,0,ADC #,10
2920 DATAROR A,0,,,JMP,35,ADC,30,ROR,30,
     ,,BVS,40
2930 DATAADC,24,,,,,,,ADC,21
2940 DATAROR,21,,,SEI,0,ADC,32,,,,,,,ADC
     ,31
2950 DATAROR,31,,,,,STA,23,,,,
2960 DATASTY,20,STA,20,STX,20,,,DEY,0,,,
     TXA,0
2970 DATA,,STY,30,STA,30,STX,30,,,BCC,40
2980 DATASTA,24,,,,,STY,21,STA,21
2990 DATASTX,22,,,TYA,0,STA,32
3000 DATATXS,0,,,,,STA,31,,,,,LDY #,10,L
     DA,23
```

```
3010 DATALDX #,10,,,LDY,20,LDA,20
3020 DATALDX,20,,,TAY,0,LDA #,10,TAX,0,,
     ,LDY,30
3030 DATALDA,30,LDX,30,,,BCS,40,LDA,24
3040 DATA,,,,LDY,21,LDA,21,LDX,22
3050 DATA,,CLV,0,LDA,32,TSX,0,,
3060 DATALDY,31,LDA,31,LDX,32,,,CPY #,10
3070 DATACMP,23,,,,,CPY,20
3080 DATACMP,20,DEC,20,,,INY,0,CMP #,10,
     DEX,0,,
3090 DATACPY,30,CMP,30,DEC,30,,,BNE,40
3100 DATACMP,24,,,,,,,CMP,21,DEC,21,,
3110 DATACLD,0,CMP,32,,,,,,,CMP,31
3120 DATADEC,31,,,CPX #,10,SBC,23,,,,
3130 DATACPX,20,SBC,20,INC,20,,
3140 DATAINX,0,SBC #,10,NOP,0,,,CPX,30,S
     BC,30,INC,30,,
3150 DATABEQ,40,SBC,24,,,,,,
3160 DATASBC,21,INC,21,,,SED,0,SBC,32,,,
     ,,,
3170 DATA SBC,31,INC,31,,
```

① *Listato del programma LEGGIFILE.*

Questo breve programma permette di leggere un file non chiuso correttamente. Per ogni byte verrà visualizzato sullo schermo il numero d'ordine e la rappresentazione (se il byte non è un carattere stampabile avremo l'indicazione del suo codice ASCII). Lo scopo è individuare il numero d'ordine dell'ultimo byte valido per poter recuperare il file con il programma di figura 2.

①

```
1000 REM ************************
1100 REM ***    LEGGIFILE:    ***
1200 REM *** VISUALIZZAZIONE ***
1300 REM *** FILE NON CHIUSO ***
1400 REM ************************
1500 :
1600 OPEN15,8,15:REM085
1700 INPUT"NOME DEL FILE,TIPO";N$,T$:REM
     377
1800 PRINT"PREMI UN TASTO QUANDO HAI IND
     IVIDUATO":REM287
1900 PRINT"LA FINE DEL FILE DA RECUPERAR
     E.":REM480
2000 OPEN1,8,2,N$+","+T$+",M":REM171
2100 GOSUB3100:REM007
2200 N=1:REM128
2300 GET#1,A$:GOSUB3100:REM264
2400 A=ASC(A$+CHR$(0)):REM126
2500 PRINTN;TAB(6);:REM118
2600 IFA<32OR(A>127ANDA<161)THENPRINT"AS
     CII="A:GOTO2800:REM080
2700 PRINTA$:REM443
2800 GOSUB3100:REM014
2900 GETT$:IFT$=""THEN N=N+1:GOTO2300:RE
     M129
3000 CLOSE1:CLOSE15:END:REM145
3100 :
3200 REM *** LETTURA ERRORE DRIVE ***
3300 :
3400 INPUT#15,ER,ER$,TR,SC:REM286
3500 IFER<20THENRETURN:REM155
3600 PRINTER;ER$;TR;SC:CLOSE1:CLOSE15:EN
     D:REM413
```

② *Listato del programma RECUPERAFILE.*

Una volta individuato (usando il programma di figura 1) l'ultimo byte valido del file si prenda nota del suo numero d'ordine e si faccia uso di questo programma per recuperare il file copiandone i dati validi in un nuovo file che (se tutto va bene!) verrà chiuso correttamente.

```
1000 REM ************************            ②
1100 REM ***   RECUPERAFILE:    ***
1200 REM ***     RECUPERO       ***
1300 REM *** FILE NON CHIUSO ***
1400 REM ************************
1500 :
1600 OPEN15,8,15:REM085
1700 INPUT"FILE DA RECUPERARE,TIPO=";N$,
     T$:REM293
1800 INPUT"FILE DI RIMPIAZZO=";N1$:REM29
     8
1900 OPEN2,8,3,N1$+","+T$+",W":REM240
2000 GOSUB3100:REM006
2100 OPEN1,8,2,N$+","+T$+",M":REM428
2200 GOSUB3100:REM008
2300 INPUT"ULTIMO BYTE DA RECUPERARE=";U
     B:REM041
2400 FORI=1TOUB:REM414
2500 PRINTI:PRINT"{CUR.SU}";:REM116
2600 GET#1,A$:REM141
2700 IFA$=""THENA$=CHR$(0):REM145
2800 PRINT#2,A$;:REM120
2900 NEXT:REM010
3000 CLOSE2:CLOSE1:CLOSE15:END:REM115
3100 :
3200 REM *** LETTURA ERRORE DRIVE ***
3300 :
3400 INPUT#15,ER,ER$,TR,SC:REM286
3500 IFER<20THENRETURN:REM155
3600 PRINTER;ER$;TR;SC:CLOSE2:CLOSE1:CLO
     SE15:END:REM383
```

③ *Mappa di memoria dei drive 1541 e 2031LP.*

Questi due modelli, pur avendo interfacce differenti (seriale il primo, parallela IEEE-488 il secondo), sono molti simili. Le principali differenze sono il collegamento delle porte del VIA 6522 #1 (quello mappato da $1800) ai connettori di interfaccia (diversi per i due drive), nonché naturalmente le routine del DOS che gestiscono l'interfaccia stessa.
In entrambi i casi abbiamo cinque buffer, di cui uno (il l#4, da $0700) è dedicato a contenere la RAM del disco.

⑤ *Esempio di dump su stampante.*
È stato chiesto a DOS-INSPECTOR di mostrarci l'inizio della pagina zero della memoria del 1541.

```
0000  01 01 01 0F 01 00 07 0F            ⑤
0008  12 04 07 03 12 00 12 00
0010  00 00 42 57 00 00 42 57      BW  BW
0018  07 03 11 00 00 00 10 00
0020  00 00 07 00 52 56 B5 4D       RVI M
0028  57 7D DD 25 29 4A 00 05   W= |%)J
0030  00 05 0A 00 FF 00 FF 00        π π
0038  07 08 82 00 00 00 FF 02          π
0040  07 02 FF 15 80 00 00 07      π
0048  00 39 00 5A 00 03 01 BA   9 Z     ┘
0050  00 FF 50 82 00 FF 0F 0A   πP  π
0058  09 12 0A 0A 0A 06 04 04
```

⑥ *Esempio di disassemblaggio su stampante.*
DOS-INSPECTOR ha disassemblato la ROM del 1541 a partire da $C100; è la routine che accende il LED del drive.

```
49408 $C100   78          SEI           ⑥
49409 $C101   A9 F7       LDA # $F7
49411 $C103   2D 00 1C    AND $1C00
49414 $C106   48          PHA
49415 $C107   A5 7F       LDA $7F
49417 $C109   F0 05       BEQ $C110
49419 $C10B   68          PLA
49420 $C10C   09 00       ORA # $00
49422 $C10E   D0 03       BNE $C113
49424 $C110   68          PLA
49425 $C111   09 08       ORA # $08
49427 $C113   8D 00 1C    STA $1C00
49430 $C116   58          CLI
49431 $C117   60          RTS
```

⑦ *Modifiche a DOS-INSPECTOR.*
Sostituendo alle linee originali quelle in figura, otterremo un programma che indaga nella memoria del computer anziché in quella del drive.

```
1460 POKEMC,PC:REM329
1470 IFPEEK(MC)<>PCTHENPRINT"?{CUR.SIN}"
     ;:REM389
1480 MC=MC+1:REM390
1490 PRINT"{3 CUR.DES}";:REM305
1500 NEXT:REM005
2140 ND=PEEK(PC):REM415
2150 RETURN:REM424
2160 REM
2170 REM                                 ⑦
```

```
Mappa di memoria dei drive 1541 e 2031LP   ③
$0000 - $02FF: pagina zero,
stack ed altre locazioni a
disposizione del DOS.
$0300 - $03FF: buffer #0
$0400 - $04FF: buffer #1
$0500 - $05FF: buffer #2
$0600 - $06FF: buffer #3
$0700 - $07FF: buffer #4
(riservato alla BAM)
$0800 - $17FF: vuoto
$1800 - $180F: VIA 6522 #1: collegato alle linee di
interfaccia
$1810 - $1BFF: immagini VIA #1
$1C00 - $1C0F: VIA 6522 #2: collegato ai motori e alla testa
di lettura/scrittura
$1C10 - $1FFF: immagini VIA #2
$2000 - $7FFF: immagini $0000/$1FFF
$8000 - $BFFF: immagini ROM
$C000 - $FFFF: ROM contenente il DOS
```

IL SALVATAGGIO FRAZIONATO DEI PROGRAMMI

di P. Opkins
Trad. e adatt. di S. Albarelli

I lunghi programmi sono spesso divisi in molte parti e caricati poi da un piccolo programma detto caricatore (Boot).
Questo articolo spiega come funziona questa tecnica e comprende una semplice dimostrazione.
Il programma dimostrativo gira su tutti i modelli Commodore in commercio, compreso il 128 (nel modo 64), e richiede l'utilizzo del drive per floppy disk.
Molti programmi complessi, specialmente programmi commerciali e pacchetti software, appaiono sul disco o sul nastro come una collezione di file.
Il programma è diviso quindi in tanti piccoli pezzi, e ogni file è un pezzo del programma.
Al primo programma, detto Boot, è affidato il compito di caricare tutti gli altri e ridurli a un unico programma.
Quando un programma è presente in questa forma su un disco, caricando la directory, essa apparirà in questo modo, essenzialmente:
Boot (caricatore)
+ Schermo del gioco
+ Musica del gioco
+ Sprite del gioco
+ L.M. del gioco
+ Programma principale.
In questo caso, facendo girare il Boot, esso caricherà tutte le parti del gioco e, dopo di ciò, spesso, si cancellerà dalla memoria.
Notare come i nomi dei pezzi del gioco inizino con un simbolo di somma per indicare che essi non devono essere caricati direttamente, poiché è il Boot che li carica per voi.
A volte, però, i programmatori dei giochi commerciali non ricorrono a questo accorgimento, perciò se caricando la directory di un programma commerciale essa non appare in questo formato, non c'è nulla di strano: sarà sufficiente caricare il primo programma del disco con un LOAD"*",8,1, e farlo girare, per vedere se ha inizio il caricamento degli altri file.
Se anche in questo caso non accade nulla è necessario leggere le istruzioni del programma in questione.

UNA PICCOLA STORIA

I primi computer che l'uomo ha creato, non disponevano ancora della ROM. La meravigliosa ROM, nella quale ora i computer trovano tutti i dati e il sistema operativo per poter funzionare, non esisteva ancora.
Allora il computer aveva bisogno di un piccolo programma che caricasse il sistema operativo in memoria.
Per inserire tale programma si ricorreva a microswitch, ognuno dei quali inseriva un bit del programma nella memoria del computer, altre volte si ricorreva a schede perforate.
Qualunque fosse il metodo, una cosa era sicura: il programma caricatore doveva essere necessariamente molto corto, quanto basta a per caricare un file in memoria.
Allora, il computer appena acceso, provvedeva al caricamento del sistema operativo in memoria.
Questo procedimento può essere simulato con un programma che carichi in memoria tutti i dati necessari, a seconda delle necessità.
Questo tipo di caricamento apre nuove prospettive alla programmazione.
Per esempio un piccolo Boot può caricare in memoria le varie parti di un gioco separatamente, in modo che esse vadano a posizionarsi in aree di memoria diverse.
Inoltre è più facile creare un gioco programmandone e creandone le diverse parti separatamente, poiché esse sono di natura diversa (un brano musicale non ha nulla e che vedere con uno schermo in alta risoluzione).
Inoltre il Boot può essere utile anche durante l'esecuzione di un programma.
Se per esempio in un gioco è necessario visualizzare sullo schermo alternativamente, molte schermate, e la memoria non è sufficiente per contenerle tutte, basta creare un Boot che, a seconda delle necessità, carichi da disco l'una o l'altra schermata.
Esistono inoltre alcuni programmi che funzionano solo se in memoria è presente una utility specifica.
Per poterli utilizzare bisogna allora caricare l'utility, farla girare e caricare il programma.
Questo lavoro può essere eliminato da un Boot che svolga questa funzione al vostro posto.

UNA SEMPLICE DIMOSTRAZIONE

Ora scriveremo insieme un piccolo programma che usi la tecnica sopradescritta. Questo programma svolge una semplice funzione: legge un file sequenziale da disco e lo visualizza sullo schermo.
Se non avete alcun file sequenziale sui vostri dischi, potete crearne uno battendo in modo diretto queste istruzioni:

```
OPEN 8,8,8,"0:XFILE,S,W,"
PRINT#8,"CIAO A TUTTI"
PRINT#8,"DALLA JACKSON"
CLOSE 8
```

Ora che avete un file sequenziale sul vostro disco, siamo pronti per creare il nostro programma.
Ecco cosa faremo: porremo nella memoria un programma BASIC che fungerà da programma principale.
In un'altra area (il buffer cassetta) porremo una routine in linguaggio macchina, che legge il file sequenziale e lo visualizza.

Infine abbiamo bisogno di un programma Boot che carichi il tutto in memoria: infatti il programma BASIC e il linguaggio macchina vanno caricati in memoria separatamente, e il Boot lo farà per noi.

LA ROUTINE IN LINGUAGGIO MACCHINA

Prima di tutto dobbiamo salvare sul disco la routine in linguaggio macchina. Il programma che ora scriveremo non è la routine in linguaggio macchina, ma è un programma che crea tale routine e la salva automaticamente.
È necessario scrivere il programma è salvarlo (in caso di futuri utilizzi), dopodiché si può inserire il disco dove va salvata la routine e far girare il programma battendo RUN e premendo RETURN.
Allora il drive comincerà a girare e il programma salverà la routine il L.M. su disco sotto il nome di "+ML".
Se il computer stampa il messaggio "ERRORE" vuol dire che avete commesso un errore nel battere il programma.
Se ciò accadesse cancellate dal disco la routine che il computer ha appena salvato su disco digitando in modo diretto:

```
OPEN15,8,15,"SO:+ML":
CLOSE15
```

Quindi caricate il programma generatore, che avete salvato, correggete gli errori, e fatelo girare nuovamente.
Vi ricordiamo che se utilizzate un 128, è necessario far girare tutti i programmi quando esso è commutato in modo 64.
Ecco il programma che genera la routine in L.M.; copiatelo molto attentamente:

```
100 DATA60,3
110 DATA162,1
120 DATA32,198,255
130 DATA32,228,255
140 DATA32,210,255
150 DATA166,144
160 DATA240,246
170 DATA76,204,255
180 OPEN 4,8,4,"0:+ML,P,W"
190 FORJ=1TO20
200 READX
210 T=T+X
220 PRINT#4,CHR$(X);
230 NEXTJ
```

ARTICOLI

```
240 CLOSE4
250 IFT<> 3054 THEN PRINT "ERRORE"
```

CREIAMO IL PROGRAMMA PRINCIPALE

Il programma Basic è molto corto. Digitare NEW seguito da RETURN, e battete questo programma.

```
100 PRINT" NOME DEL FILE":INPUTN$
110 OPEN 1,8,2,N$
120 SYS828
130 CLOSE1
```

Ora salvate questo programma sullo stesso disco dove avete salvato il linguaggio macchina digitando in modo diretto:
SAVE"0:+BASIC",8
non fate girare questo programma, poiché esso dovrà essere caricato dal Boot per poter funzionare correttamente.

CREIAMO IL BOOT

Digitate nuovamente NEW seguito da RETURN.
Il Boot varia lievemente a seconda del computer che si utilizza.
Digitare la linea 100 adatta al vostro computer.
Per VIC 20, C64 e 128 (in modo 64):
100 DATA144,198,631
Per C16 e Plus/4:
100 DATA144,239,1319
I tre numeri alla linea 100 rappresentano le locazioni della variabile di status del computer (ST), il contatore del buffer di tastiera, e il buffer della tastiera.
Il primo valore fa si che il programma funzioni su tutti i modelli previsti, e gli altri due sono utilizzati per far si che il programma scriva sullo schermo. Ecco altre linee da aggiungere al programma:

```
110 IFX=1GOTO200
120 X=1
130 LOAD"+ML",8,1
140 STOP
```

Qui abbiamo usato la tecnica dell'overlay.
La linea 140 non viene mai eseguita, poiché il programma riparte dopo aver caricato il file "+LM".
Allora trova le variabili intatte, con gli stessi valori che avevano prima che il load fosse eseguito.
Quindi arrivato alla linea 110, esegue un salto alla linea 200, che scriveremo ora:

```
200 READ A,B,C
210 POKE 840,A
```

Tra poco scriveremo le linee che caricano in memoria il programma BASIC principale.
Per fare ciò bisogna utilizzare un trucco che, in caso si carichi il linguaggio macchina, non è necessario.
Questa tecnica che si rende necessaria, consiste nel far eseguire al computer delle istruzioni, semplicemente inserendo dei caratteri nel buffer tastiera. Ecco le linee che utilizzano tale trucco:

```
220 D$=CHR$(17)
230 R$=CHR$ (147) +D$ +D$ +D$ +"LOAD"
240 N$=CHR$(34)+"+BASIC" +CHR$(34)
250 PRINT R$+N$+",8" ++D$+D$
260 PRINT D$+D$+"RUN" +CHR$(19)
270 POKEB,2:POKEC,13: POKEC+1,13
```

Se non avete mai usato questa utile tecnica, queste linee vi appariranno confuse.
Brevemente, queste linee dicono al computer di scrivere sullo schermo due comandi per noi.
Esse appaiono quando il programma gira:
LOAD"+BASIC",8
RUN
I comandi sono posti alla terza e all'ottava linea dello schermo.
Se voi premeste RETURN due volte, dopo aver posto il cursore nella posizione di HOME, le istruzioni caricherebbero il programma "+ BASIC" e lo farebbero partire.
Ma mettendo nel buffer della tastiera due simboli RETURN, si ottiene lo stesso effetto, solo che non c'è bisogno di battere RETURN, poiché è il computer stesso che lo fa per noi.
Ora il nostro programma di Boot è completo.
Basta salvarlo su disco con il nome di Boot.

Siate sicuri di aver salvato una copia del programma su disco, prima di farlo girare, perché andrebbe irrimediabilmente cancellato dalla memoria del computer.
Dopo aver salvato il Boot sul disco, leggendo la directory, dovrebbero apparirvi questi nomi di files:
Boot (il Boot appena battuto)
+ML (la routine il l.m.)
+BASIC (il programma principale)
XFILE (il file creato che leggeremo)
Se caricate e fate girare il programma Boot, esso caricherà prima il linguaggio macchina, e successivamente il programma principale.
Allora, partito il programma principale, vi sarà chiesto il nome del file da leggere e visualizzare su video.
Nel nostro caso, abbiamo creato il file "XFILE", e possiamo leggerlo, come possiamo fare con qualsiasi file sequenziale.
Dopo aver letto il file, il programma si fermerà, ma basterà un semplice RUN per farlo ripartire e poter leggere altri file.
Utilizzando questa curiosa e utile tecnica potrete rendere più belli e pratici i vostri programmi, e creare degli effettivi pacchetti software!

ARTICOLI

LE MANI SUL VIDEO

di: G. Ferri

Oltre ai comuni terminali esistono i cosiddetti "touch screen" cioè quegli apparecchi con lo schermo sensibile al tocco. Vediamo di che cosa si tratta e come funzionano.
Con il termine "touch screen" si denotano quei terminali che sono sensibili al tocco dell'utente sullo schermo.
Com'è noto l'organo principale e più usato di input è la tastiera, ma alcuni costruttori hanno pensato di utilizzare anche lo schermo per l'immissione dei dati.
Questa tecnica è nata molti anni fa e fu inizialmente usata laddove l'uso della tastiera come strumento di ingresso dati appariva troppo complicato: per esempio in certe fabbriche americane questa soluzione ha trovato larga diffusione, anche in applicazioni di gestione.
Ci sono diverse tecnologie di schermi tattili (così vengono tradotti in italiano): quella ottica, l'acustica, quella a resistenza di membrana e la capacitativa.
Negli USA ci sono varie ditte che producono "touch screen" e una delle più importanti è senz'altro la AT&T Information System che ha adottato il sistema capacitativo.
La casa che comunque a contribuito maggiormente a far conoscere i "touch screen" al grosso pubblico è stata la Hewlett-Packard con il suo personal HP 150.
Le varie tecnologie sono diverse tra di loro e possono anche influenzare le modalità di uso dello schermo stesso.
Nel caso delle prime tre tecniche il comando viene accettato sia nel caso si usi il dito che la penna. I sistemi capacitativi invece chiedono che il comando venga dato con il dito con una opportuna carica elettrica.
Inoltre, negli schermi realizzati secondo le prime due tecnologie è sufficiente sfiorare lo schermo anziché toccarlo fisicamente perché il comando sia recepito.
Un altro elemento molto importante che differenzia le varie tecnologie e gli stessi schermi tattili, realizzati secondo la stessa tecnica, è il grado di risoluzione che permette di definire le aree che identificano le varie funzionalità.
Si va da un minimo di una trentina di punti di sensibilità come si trova nei sistemi più semplici, fino ad arrivare a oltre un milione nei megasistemi super evoluti.
L'uso del touch screen nei personal computer rappresenta un grosso stimolo per gli utenti: secondo alcuni esperti lo schermo tattile abbinato con la tastiera rendono il sistema di più facile utilizzo.
Ed effettivamente si può dire che così è stato, soprattutto negli Stati Uniti: tanto per fare un esempio, alcune banche hanno installato in certe loro filiali dei terminali con schermo sensibile. Il cliente può verificare l'andamento degli investimenti, chiedere informazioni su azioni o altri titoli di suo interesse e altre operazioni ancora.
L'utilizzo del computer in questa forma facilita senz'altro il colloquio utente-macchina poiché evita perdite di tempo ed è quasi impossibile sbagliare (a meno che non lo si voglia).
Come ultima cosa, pensiamo che si possa vedere il "touch screen" quasi come il padre del mouse.
Esso, infatti, rappresenta il primo tentativo di superamento dei tradizionali sistemi di input. Ciò non vuole certo dire che questi vadano sostituiti oppure che bisogna cercare a tutti i costi di non usare la tastiera.
Un conto è infatti cambiare un oggetto che decade per prestazioni, di fronte a ciò che tecnicamente lo supera (vedi lettore di schede, ma anche i "personal-utenti" si accorgono di come i computer che rimangono legati alla cassetta vengono soppiantati da quelli che permettono l'utilizzo del floppy/disk).
Altro, invece, è cercare un utilizzo il più intelligente e comodo possibile dei mezzi che si hanno a disposizione.
È molto meno realizzabile un uso di servizi bancari tramite mouse che non tramite schermo sensibile; viceversa, dato che il "touch screen" rivolto al grande pubblico si presenta con un numero limitato di campi selezionabili, laddove tale numero cresce è molto meglio utilizzare il mouse.
Perciò i terminali con lo schermo sensibile hanno ancora la loro ragione di esistere; se è pur vero che per certe applicazioni, soprattutto grafiche, sono stati superati da altri strumenti (appunto mouse, joystick, penna ottica) è altrettanto sicuro che possono

svolgere con successo compiti in cui non è necessario il massimo dettaglio o l'accuratezza nell'ingresso dei dati.

I NUMERI DEI DISPOSITIVI

di M.S. Tomczyk
Trad. e adatt. di M. Anticoli

Vediamo come usare al meglio la tastiera, la stampante e il disk drive conoscendo i numeri dei dispositivi.
Per mandare delle informazioni alla stampante, al disk drive, al modem o perfino al video, bisogna aprire un canale usando un comando OPEN e il proprio numero di dispositivo.
Qui in basso riportiamo i numeri più comuni di dispositivi usati dai computer Commodore:

DISPOSITIVO	NUMERO
Tastiera	*0*
Registratore (cassetta)	*1*
Modem	*2*
Video	*3*
Stampante	*4-7*
Disk drive	*8-11*

Prima di comunicare con un particolare dispositivo si deve aprire un file al dispositivo.
In un comando come OPEN 1,4 il primo numero è il numero del file e il secondo è il numero del dispositivo (dispositivo 4 è la stampante).
Il comando PRINT# è usato per mandare dei caratteri a un particolare dispositivo.
Per esempio: PRINT#1, "CIAO" 'dice' alla stampante di stampare la parola ciao (naturalmente prima di eseguire questo comando bisogna usare il comando OPEN).
Quando si finisce di usare il dispositivo (nel nostro caso la stampante) bisogna chiudere il file con un comando CLOSE 1.

COME USARE I NUMERI DI DISPOSITIVI CON IL NASTRO O IL DISCO

Probabilmente già si conosce come salvare o caricare un programma da nastro o da disco.
I formati standard sono mostrati qui sotto:
Salvare su nastro: SAVE "nomefile"
Caricare da nastro: LOAD "nomefile"
Salvare su disco: SAVE "nomefile", 8
Caricare da disco: SAVE "nomefile", 8
Caricare la directory da disco: LOAD"$",8 (poi digitare LIST e premere RETURN).
Il numero di dispositivo per il nastro è 1, ma se si digita un comando di LOAD o di SAVE senza numero di dispositivo alla fine, il computer automaticamente (il famoso modo di 'default') presume che si usi il nastro.
Questo avviene perché il nastro è più popolare che il disco.
Se si vuole caricare un programma in linguaggio macchina si deve usare un ,1 alla fine del comando; bisogna fare questo così si è sicuri di rilocare il programma nella locazione di memoria originale. Per esempio vogliamo caricare un programma in linguaggio macchina, si usa: LOAD "nome file", 8,1
È anche possibile collegare insieme più disk drive.
Molti programmatori esperti collegano differenti disk drive al loro computer e danno i numeri di dispositivo da 8 a 11 (per vedere come cambiare numero di dispositivo bisogna consultare la guida di riferimento del disk drive).
Riassumendo, se si usa il nastro per caricare o salvare un programma si può anche omettere il numero di dispositivo se invece si usa il disco bisogna mettere il numero di dispositivo (8), se si carica un programma in linguaggio macchina bisogna mettere un ,1 alla fine del comando.
Se si usa il nastro per caricare un programma in linguaggio macchina è consigliabile mettere il numero di dispositivo e il ,1 finale e non andare in modo di default, per esempio:
LOAD "nome file",1,1
I nuovissimi computer Commodore (128, PLUS/4 e C16) contengono nel loro basic degli speciali comandi per il disco come:
DLOAD (carica un programma da disco).
Questi comandi non richiedono numero di dispositivo, benché tutti i computer Commodore lavorino con i comandi mostrati nella cartina in alto.
Il numero di dispositivi per il modem, cioè per effettuare delle comunicazioni fra computer, è il 2, ma non approfondiamo l'argomento poiché tale periferica è poco usata dagli utenti italiani dei computer Commodore.

LEGGERE DA TASTIERA

Per immettere dei dati da tastiera si usa normalmente il comando di INPUT.
Un'altra via è quella di aprire un file usando il numero di dispositivo 0 e usando il comando INPUT#.
Quest'ultima via è migliore poiché elimina il punto interrogativo che appare automaticamente quando si usa il comando INPUT.
Per esempio, si deve digitare un elenco e si vuole eliminare il punto interrogativo e si vuole mettere invece un segno di dollaro.
Ecco qui sotto l'esempio:

```
10 OPEN 1,0
20 PRINT CHR$ (147) "CALCO
LO DELLE SPESE":PRINT
30 PRINT "AFFITTO....$";
40 INPUT#1,A:PRINT
50 PRINT "CIBO... $";
60 INPUT#1,B:CLOSE 1:PRINT
70 PRINT:PRINT"PER IL CIBO E
PER L'AFFITTO"
80 PRINT"PAGATE $"A+B
90 PRINT "AL MESE."
```

La linea 10 apre il file 1 sulla tastiera (dispositivo 0).
La linea 20 usa CHR$ (147) per pulire il video, poi stampa il titolo e una linea bianca.

La linea 30 è la richiesta dell'affitto mensile.
La linea 40 usa INPUT#1 invece del più familiare comando di INPUT.
INPUT#1 visualizza il numero appena digitato, alla fine della istruzione di PRINT (nella linea 30) ma senza visualizzare il punto interrogativo.
La linea 50 è la richiesta del costo del cibo.
La linea 60 usa il comando INPUT# e mette il numero digitato dall'utente nella variabile B.
Poi chiude il file da tastiera.
Le linee 70, 80 e 90 stampano dei messaggi, bisogna notare la linea 80 dove addizioniamo le variabili A e B.

MANDARE DELLE INFORMAZIONI ALLA STAMPANTE

Per aprire un canale in una stampante bisogna digitare OPEN 4,4 dove il primo 4 rappresenta il numero del file e il secondo il numero di dispositivo.
Per stampare su carta bisogna usare il comando PRINT#4 seguito da una virgola e da informazioni o variabili.
PRINT#4 lavora come un comando PRINT eccetto che l'informazione è mandata alla stampante invece che al video.

ARTICOLI

Ecco un piccolo esempio:

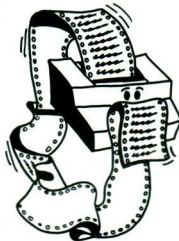
```
10 PRINT CHR$(147) "VISUALIZ
ZO DELLE INFORMAZIONI SUL
LO SCHERMO."
20 OPEN4,4:PRINT#4,"PRINT#4
MANDA LE INFORMAZIONI SU
STAMPANTE.": CLOSE4
```

La linea 10 usa un comando PRINT per visualizzare su video un messaggio e CHR$(147) pulisce il video.
La linea 20 apre il file 4 e il dispositivo 4 (stampante), poi usa un comando PRINT#4 seguito da una virgola e dalle informazioni, così il messaggio è stampato su carta, infine chiude il file.

UN SEMPLICE PROGRAMMA

Ecco qui in basso un semplicissimo programma; si deve immettere un numero, questo numero viene addizionato al precedente (naturalmente all'inizio viene addizionato a zero), il numero immesso e la somma dell'addizione vengono visualizzati su video e su stampante.

```
10 OPEN 4,4
20 PRINT CHR$(147) "PRO
GRAMMA SEMPLICE":
PRINT:PRINT
30 PRINT"INTRODUCI UN NU
MERO":PRINT
40 PRINT"ADDIZIONO";:
INPUT A:PRINT#4,
"ADDIZIONO"A
50 PRINT"TOTALE"A+B:
PRINT#4,
CHR$(18)"TOTALE"CHRS$(146)
A+B
60 B=A+B
70 GOTO 40
```

Per fermare questo programma bisogna premere insieme i tasti RUN/STOP e RESTORE.

DIREZIONE, REDAZIONE E AMMINISTRAZIONE
Via Rosellini, 12 - 20124 Milano
Telefoni: (02) 68.03.68 - 68.00.54
68.80.951-2-3-4-5
Telex 333436 GEJ IT

SEDE LEGALE:
Via G. Pozzone, 55 - 20121 Milano

DIRETTORE RESPONSABILE:
Giampietro Zanga

COORDINAMENTO EDITORIALE:
Angelo Cattaneo
Luca Zaninello

GRAFICA E IMPAGINAZIONE:
Wilma Germani

FOTOCOMPOSIZIONE:
GDB fotocomposizione
Via Tagliamento, 4 - Milano
Tel. 56.92.110 - 53.92.546

STAMPA:
Grafika 78 - Pioltello - Milano

AUTORIZZAZIONE ALLA PUBBLICAZIONE:
Numero in attesa di autorizzazione

Per la rivista non è prevista la sottoscrizione di abbonamenti

PUBBLICITÀ
Concessionario per l'Italia e l'Estero
J. Advertising s.r.l.
V.le Restelli, 5 - 20124 MILANO
Tel. (02)
68.82.895-68.80.606-68.87.233
Tlx 316213 REINA I

Concessionario esclusivo per la diffusione in Italia e Estero:
SODIP - Via Zuretti, 25
20125 MILANO

Spedizione in abbonamento postale
Gruppo III/70
Prezzo della rivista L. 8.000
Numeri arretrati L. 16.000

ESPRIMI IL TUO TALENTO

- DIECI VIDEOLEZIONI (fascicoli con cassetta) SOFTWARE PER C64/128 E C64 PERSONAL COMPUTER
- DA RILEGARE IN UNO SPLENDIDO VOLUME
- IN EDICOLA OGNI 15 GIORNI A L. 8.000

CON LA 1ª LEZIONE LA FAVOLOSA SCACCHIERA ELETTRONICA

A SCUOLA DI SCACCHI
L. 8.000 1
COMMODORE C64/128
C64 PERSONAL COMPUTER
I GRANDI CAMPIONI:
Bobby Fischer
STORIA, TEORIA E PRATICA PER PRINCIPIANTI ED ESPERTI

A SCUOLA DI SCACCHI
1ª LEZIONE
più la favolosa scacchiera elettronica
GRUPPO EDITORIALE JACKSON
DIVISIONE GRANDI OPERE

La prima grande opera a fascicoli con cassetta software per imparare in modo interattivo i segreti del millenario gioco degli scacchi.

A SCUOLA DI SCACCHI

STORIA, TEORIA E PRATICA PER PRINCIPIANTI ED ESPERTI

Chi sono veramente Karpov e Kasparov?
Come è nato e come si è diffuso nei secoli questo nobile gioco?
Come riuscire a vincere tutte le partite?
Scopri tutti i segreti dei grandi campioni, le loro mosse più abili e famose e le strategie di gioco.
Un'opera rivoluzionaria da leggere, da consultare, da giocare.

Per esaltare le tue capacità artistiche per imparare a capire tutte le applicazioni possibili di grafica con il computer.

CORSO DI grafica 64 128

STRUMENTI, APPLICAZIONI E IMMAGINI COL COMPUTER

Troverai tutti i consigli, i trucchi, i suggerimenti e gli aiuti necessari per sfruttare le potenzialità grafiche del computer in modo nuovo e originale. Dall'architettura all'abbigliamento, dalle auto al mondo dello spettacolo, non c'è settore in cui la computer graphics non sia applicata, prova anche tu con il tuo Commodore.

- DIECI LEZIONI (fascicoli con cassetta) SOFTWARE PER C64/128 E C64 PERSONAL COMPUTER
- DA RILEGARE IN UNO SPLENDIDO VOLUME
- IN EDICOLA OGNI 15 GIORNI A L. 8.000

IN EDICOLA DUE MOSSE VINCENTI JACKSON